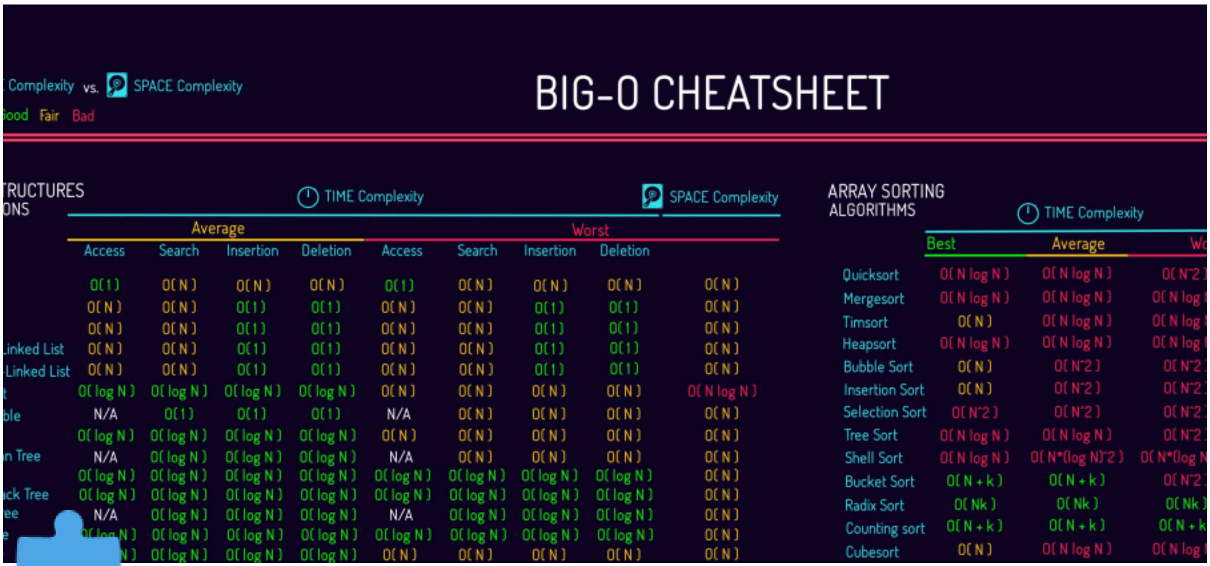


Module 1 | Introduction to Algorithm Analysis

Importance	⭐
Syllabus	Basic Algorithmic Analysis: Asymptotic notations: Big O, little o, omega, and theta notations, Running time calculations, Identifying differences among best, average, and worst case behaviors; Complexity analysis- Time and space

Time Complexity

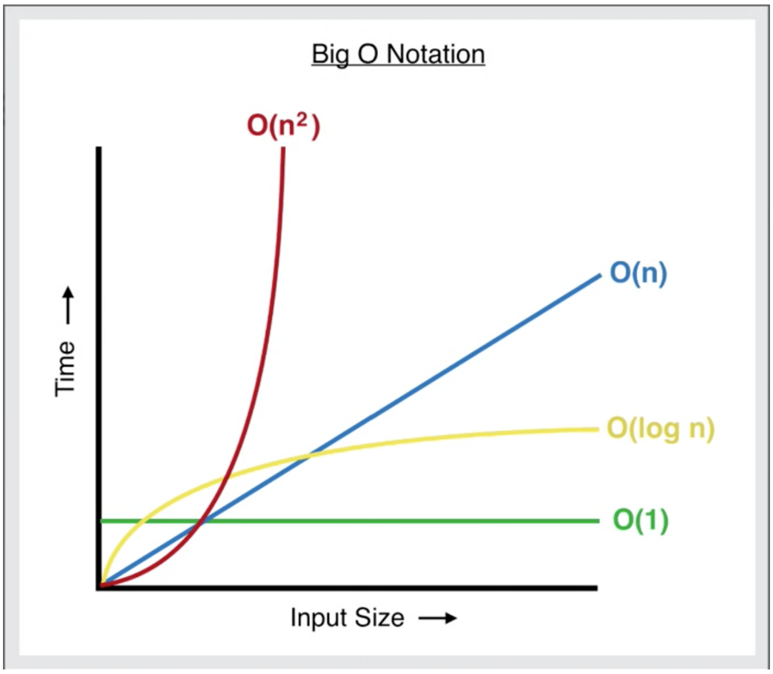
Time complexity is the study of efficiency of algorithms.

Big O Notation

Let the time complexity of an algorithm be $k_1n^2 + k_2n + 36$ then its complexity in terms of 'O' will be the highest order term. i.e O(n^2)

For constants the time complexity is O(1)

Big O in graph



Asymptotic notations

1. Big oh notation (O)
2. Little oh notation (o)
3. Big omega notation (Ω)
4. Big theta notation (θ)

1. Big oh Notation (O)

Big oh is used to describe asymptotic upper bound.

Mathematically, if f(x) describe running time of an algorithm; f(x) is O(g(x)) iff there exist positive constants c and n_0 such that **0≤f(x)≤cg(x)** for all **n≥**n_0

2. Little oh Notation (o)

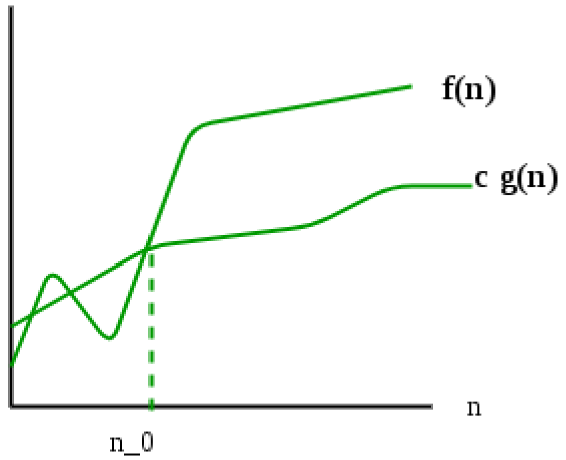
Little oh means loose upper-bound of f(x). Little oh is a rough estimate of the maximum order of growth whereas Big oh maybe the actual order of growth

Mathematically, f(x)=o(g(x)) means $lim(n \to \infty)f(n)/g(n) = 0$

3. Big omega notation (Ω)

Just like O provides an asymptotic upper bound, Ω provides asymptotic lower bound .

Mathematically, if f(x) describe running time of an algorithm; f(x) is said to be Ω (g(x)) if there exists positive constants C and n_0 such that 0≤cg(x)←f(x) for all n≥n_0



f(n) = Omega(g(n))

4. Big theta notation (θ)

Let f(x) define the running time of an algorithm, f(x) is said to be $\theta(g(x))$ iff f(x) is O(g(x)) and f(x) is Ω(g(x))

Mathematically,
0≤f(x)≤c_1g(x) for all n≥n_0 - (1)
0≤c_2g(x)≤f(x) for all n≥n_0 - (2)
(1)+(2)
0≤c_2g(x)≤f(x)≤c_2g(x) for all n≥n_0

It means there exist positive constants c_1 and c_2 such that f(x) is sandwiched between c_2g(x) and c_1g(x)

Order of Runtimes

Better→ **1 <** $logn$ **<** n **<** $nlogn$ **<** n^2 **<** n^3 **<** 2^n **<** n^n ←Worse

Space complexity

Space complexity is a function describing the amount of memory (space) an algorithm takes in terms of the amount of input to the algorithm.

Ex: Getting an array of size n has space complexity of O(n)

Auxiliary Space

Auxiliary Space is the extra space or temporary space used by an algorithm.

For CP

Acceptance Complexity by Inputs:-

Length of Input (N)	Worst Accepted Algorithm
$\leq [10..11]$	$O(N!), O(N^6)$
$\leq [15..18]$	$O(2^N * N^2)$
$\leq [18..22]$	$O(2^N * N)$
≤ 100	$O(N^4)$
≤ 400	$O(N^3)$
$\leq 2K$	$O(N^2 * \log N)$
$\leq 10K$	$O(N^2)$
$\leq 1M$	$O(N * \log N)$
$\leq 100M$	$O(N), O(\log N), O(1)$

For Competitive Programming

Online resources

- Code Chef | DSA Complexity Analysis & Basics Warmup
- Practice MCQs